WE CLAIM:

1. A method of rendering objects, the method comprising, for each said object within a scanline, the steps of:

determining each boundary pixel that overlaps both sides of a border of the object;

computing a real opacity of each said boundary pixel, wherein said real opacity of a said boundary pixel is dependent upon an intrinsic opacity of the object, winding counts for subregions of said boundary pixel, and values representative of the areas of the respective subregions with respect to the total area of the boundary pixel; and

rendering each said boundary pixel with said computed real opacity.

2. A method as claimed in claim 1, wherein said real opacity is a weighted sum of real opacities of respective said subregions, where said real opacities of respective subregions are weighted with said values, and said real opacity of a said subregion is $1 - (1 - \alpha)^{|n|}$, where α is the intrinsic opacity of the object and n is the winding count for the subregion.

3. A method as claimed in claim 1, wherein said computing step of the real opacity of each said boundary pixel, comprises the sub-steps of:

computing real opacities of a plurality of sampling points within each said boundary pixel, wherein said real opacity at a said sampling point is dependent upon the intrinsic opacity of said object and the winding count for that sampling point, and

determining the real opacity of each said boundary pixel, wherein the real opacity of a said boundary pixel is the sum of the computed real opacities at the sampling points of said boundary pixel divided by the total number of sampling points in the pixel.

4. A method as claimed in claim 1, wherein said computing step of the real opacity of each said boundary pixel, comprises the sub-steps of:

determining those areas within each said boundary pixel which have a constant winding count;

computing real opacities of a plurality of areas within each said boundary pixel, wherein said real opacity of a said area is dependent upon the intrinsic opacity of said object and the winding count for that area; and

determining the real opacity of each said boundary pixel, wherein the real opacity of a said boundary pixel is the sum of the product of percentage areas of pixel occupied by each area of said boundary pixel and its computed real opacity.

5. A method as claimed in claim 1, wherein at least one of said objects is a simple polygon.

6. A method as claimed in claim 1, wherein at least one of said objects is a self-overlapping polygon.

7. A method as claimed in claim 1, wherein for a given object of uniform intrinsic opacity, said method further comprises:

computing the real opacities for winding counts 1 to m, where m is a positive integer;

8. A method as claimed in claim 1, wherein the method further comprises the sub-steps:

determining pixels of said inside of said object other than said boundary pixels;

computing a real opacity of each said inner pixel, wherein said real opacity is dependent upon an intrinsic opacity of said object and winding count for that inner pixel; and

rendering each said inner with said determined real opacity.

9. Apparatus for rendering objects, the apparatus comprising processing means for processing each said object within a scanline, the processing means comprising:

means for determining each boundary pixel that overlaps both sides of a border of the object;

means for computing a real opacity of each said boundary pixel, wherein said real opacity of a said boundary pixel is dependent upon an intrinsic opacity of the object, winding counts for subregions of said boundary pixel, and values representative of the areas of the respective subregions with respect to the total area of the boundary pixel; and

means for rendering each said boundary pixel with said computed real opacity.

10. Apparatus as claimed in claim 9, wherein said real opacity is a weighted sum of real opacities of respective said subregions, where said real opacities of respective subregions are weighted with said values, and said real opacity of a said subregion is $1 - (1 - \alpha)^{|n|}$, where α is the intrinsic opacity of the object and n is the winding count for the subregion.

11. Apparatus as claimed in claim 9, wherein said computing means for computing the real opacity of each said boundary pixel comprises:

means for computing real opacities of a plurality of sampling points within each said boundary pixel, wherein said real opacity at a said sampling point is dependent upon the intrinsic opacity of said object and the winding count for that sampling point, and

means for determining the real opacity of each said boundary pixel, wherein the real opacity of a said boundary pixel is the sum of the computed real opacities at the sampling points of said boundary pixel divided by the total number of sampling points in the pixel.

12. Apparatus as claimed in claim 9, wherein said computing means for computing the real opacity of each said boundary pixel comprises:

means for determining those areas within each said boundary pixel which have a constant winding count;

means for computing real opacities of a plurality of areas within each said boundary pixel, wherein said real opacity of a said area is dependent upon the intrinsic opacity of said object and the winding count for that area; and

means for determining the real opacity of each said boundary pixel, wherein the real opacity of a said boundary pixel is the sum of the product of percentage areas of pixel occupied by each area of said boundary pixel and its computed real opacity.

13. Apparatus as claimed in claim 9, wherein at least one of said objects is a simple polygon.

14. Apparatus as claimed in claim 9, wherein at least one of said objects is a self-overlapping polygon.

15. Apparatus as claimed in claim 9, wherein for a given object of uniform intrinsic opacity, said apparatus further comprises:

means for computing the real opacities for winding counts 1 to m, where m is a positive integer.

16. Apparatus as claimed in claim 9, wherein the apparatus further comprises:

means for determining pixels of said inside of said object other than said boundary pixels;

means for computing a real opacity of each said inner pixel, wherein said real opacity is dependent upon an intrinsic opacity of said object and winding count for that inner pixel; and

means for rendering each said inner with said determined real opacity.

17. A computer program for rendering objects, the computer program comprising processing code for processing each said object within a scanline, the processing code comprising:

code for determining each boundary pixel that overlaps both sides of a border of the object;

code for computing a real opacity of each said boundary pixel, wherein said real opacity of a said boundary pixel is dependent upon an intrinsic opacity of the object, winding counts for subregions of said boundary pixel, and values representative of the areas of the respective subregions with respect to the total area of the boundary pixel; and

code for rendering each said boundary pixel with said computed real opacity.

18. A computer program as claimed in claim 17, wherein said real opacity is a weighted sum of real opacities of respective said subregions, where said real opacities of respective subregions are weighted with said values, and said real opacity of a said subregion is $1 - (1 - \alpha)^{|n|}$, where α is the intrinsic opacity of the object and n is the winding count for the subregion.

19. A computer program as claimed in claim 17, wherein said computing code for computing the real opacity of each said boundary pixel comprises:

code for computing real opacities of a plurality of sampling points within each said boundary pixel, wherein said real opacity at a said sampling point is dependent upon the intrinsic opacity of said object and the winding count for that sampling point, and

code for determining the real opacity of each said boundary pixel, wherein the real opacity of a said boundary pixel is the sum of the computed real opacities at the sampling points of said boundary pixel divided by the total number of sampling points in the pixel.

20. A computer program as claimed in claim 17, wherein said computing code for computing the real opacity of each said boundary pixel comprises:

code for determining those areas within each said boundary pixel which have a constant winding count;

code for computing real opacities of a plurality of areas within each said boundary pixel, wherein said real opacity of a said area is dependent upon the intrinsic opacity of said object and the winding count for that area; and

code for determining the real opacity of each said boundary pixel, wherein the real opacity of a said boundary pixel is the sum of the product of percentage areas of pixel occupied by each area of said boundary pixel and its computed real opacity.

21. A computer program as claimed in claim 17, wherein at least one of said objects is a simple polygon.

22. A computer program as claimed in claim 17, wherein at least one of said objects is a self-overlapping polygon.

23. A computer program as claimed in claim 17, wherein for a given object of uniform intrinsic opacity, said computer program further comprises:

code for computing the real opacities for winding counts 1 to m, where m is a positive integer.

24. A computer program as claimed in claim 17, wherein the computer program further comprises:

code for determining pixels of said inside of said object other than said boundary pixels;

code for computing a real opacity of each said inner pixel, wherein said real opacity is dependent upon an intrinsic opacity of said object and winding count for that inner pixel; and

code for rendering each said inner with said determined real opacity.

25. A method of rendering a self-overlapping polygon, wherein the polygon is a set of one or more closed curves each comprising line segments, and the method performing, for a currently scanned pixel that overlaps both sides of a said line segment of the self-overlapping polygon within a currently scanned scanline, the steps of:

decomposing that portion of the polygon that lies within the currently scanned pixel into a number of closed loops comprising at least those portions of those line segments that lie within the currently scanned pixel, said closed loops are such that when they are combined the combination is substantially equivalent to that portion of the polygon that lies within the currently scanned pixel ;

combining incrementally said closed loops and determining one or more winding count values representative of respective weighted averages of winding counts of said combined closed loops;

determining a real opacity of the currently scanned pixel according to a predetermined fill rule utilising an intrinsic opacity of said polygon and said one or more winding count values, and

rendering said currently scanned pixel with said determined real opacity.

26. A method as claimed in claim 25, wherein each closed curve comprises a plurality of line segments and that portion of one or more line segments of a said closed curve that lies within the currently scanned pixel is referred to as a contour segment and said decomposing step comprises the sub-step of:

creating one or more of said closed loops from respective one or more groups comprising two said contour segments that have opposing directions.

27. A method as claimed in 25, wherein each closed curve comprises a plurality of line segments and that portion of one or more line segments of a said closed curve that lies within the currently scanned pixel is referred to as a contour segment and said decomposing step comprises the sub-step of:

creating one or more of said closed loops from respective one or more groups comprising two said contour segments that have opposing directions; and

creating one or more of said closed loops from respective one or more groups comprising a single contour segment.

28. A method as claimed in claim 25, wherein each closed curve comprises a plurality of line segments and that portion of one or more line segments of a said closed curve that lies within the currently scanned pixel is referred to as a contour segment and said decomposing step comprises the sub-step of:

creating one or more of said closed loops from respective one or more groups comprising a single contour segment.

29. A method as claimed 27 or 28, wherein said sub-step of creating one or more closed loops from respective one or more groups comprising a single contour segment comprises, for each closed loop comprising a said single contour segment, the sub-steps of:

determining an area value of a region bounded by the single contour segment and pixel boundary; and

creating a binary mask representative of said region.

30. A method as claimed 26 or 27, wherein said sub-step of creating one or more closed loops from respective one or more groups comprising two contour segments comprises, for each closed loop comprising two said contour segments, the sub-steps of:

determining an area value of a first region bounded by one of said two contour segments and pixel boundary;

creating a binary mask representative of said first region;

determining an area value of a second region bounded by the other of said two contour segments and pixel boundary;

creating a binary mask representative of said second region;

determining an area value of said closed loop comprising said two contour segments corresponding to an area value of the intersection of said first and second regions utilizing said area values and binary masks of said first and second regions; and

creating a binary mask representative of said closed loop comprising said two contour segments utilizing said binary masks of said first and second regions.

31. A method as claimed in claim 25, wherein said step of combining incrementally said closed loops comprises the sub-steps of:

determining an area value corresponding to an area value of a combination of a current said closed loop and previously combined closed loops utilizing area values and binary masks of said current closed loop and previously combined closed loops; and

creating a binary mask representative of said combination utilizing said binary masks of said current closed loop and previously combined closed loops.

32. A method as claimed in claim 31, wherein said one or more weighted averages is dependent upon said area values of said combined closed loops.

33. A method as claimed in claim 25, wherein said one or more weighted averages are determined incrementally and stored in respective one or more accumulators.

34. A method as claimed in claim 25, wherein the method comprises the step of:

approximating those line segments that lie within the currently scanned pixel with one or two line segments.

35. A method as claimed in claim 25, wherein the predetermined fill rule is an odd-even fill rule.

36. A method as claimed in claim 25, wherein the predetermined fill rule is a non-zero fill rule.

37. A method as claimed in claim 25, wherein the predetermined fill rule is a winding-counting fill rule.

38. A method of rendering a self-overlapping polygon, wherein the polygon is a set of one or more closed curves each comprising line segments, and the method performing, for a currently scanned pixel that overlaps both sides of a said line segment of the self-overlapping polygon within a currently scanned scanline, the steps of:

decomposing that portion of the polygon that lies within the currently scanned pixel into a number of clockwise or counterclockwise closed loops comprising at least those portions of those line segments that lie within the currently scanned pixel, said closed loops are such that when they are combined the combination is substantially equivalent to that portion of the polygon that lies within the currently scanned pixel;

combining incrementally said clockwise and counterclockwise closed loops respectively to produce two corresponding regions, and determining two winding count values representative of respective weighted averages of winding counts of said clockwise and counterclockwise closed loops;

determining a real opacity of the currently scanned pixel according to a predetermined fill rule utilising an intrinsic opacity of said polygon and said two winding count values, and

rendering said currently scanned pixel with said determined real opacity.

39. A method as claimed in claim 38, wherein each closed curve comprises a plurality of line segments and that portion of one or more line segments of a said closed curve that lies within the currently scanned pixel is referred to as a contour segment and said decomposing step comprises the sub-step of:

creating one or more of said clockwise and counterclockwise closed loops from respective one or more groups comprising two said contour segments that have opposing directions.

40. A method as claimed in 38, wherein each closed curve comprises a plurality of line segments and that portion of one or more line segments of a said closed curve that lies within the currently scanned pixel is referred to as a contour segment and said decomposing step comprises the sub-step of:

creating one or more of said said clockwise and counterclockwise closed loops from respective one or more groups comprising two said contour segments that have opposing directions; and

creating one or more of said said clockwise and counterclockwise closed loops from respective one or more groups comprising a single contour segment.

41. A method as claimed in claim 38, wherein each closed curve comprises a plurality of line segments and that portion of one or more line segments of a said closed curve that lies within the currently scanned pixel is referred to as a contour segment and said decomposing step comprises the sub-step of:

creating one or more of said said clockwise and counterclockwise closed loops from respective one or more groups comprising a single contour segment.

42. A method as claimed 40 or 41, wherein said sub-step of creating one or more said clockwise and counterclockwise closed loops from respective one or more groups comprising a single contour segment comprises, for each said clockwise and counterclockwise closed loop comprising a said single contour segment, the sub-steps of:

determining an area value of a region bounded by the single contour segment and pixel boundary; and

creating a binary mask representative of said region.

43. A method as claimed 39 or 40, wherein said sub-step of creating one or more said clockwise and counterclockwise closed loops from respective one or more groups comprising two contour segments comprises, for each said clockwise or counterclockwise closed loop comprising two said contour segments, the sub-steps of:

determining an area value of a first region bounded by one of said two contour segments and pixel boundary;

creating a binary mask representative of said first region;

determining an area value of a second region bounded by the other of said two contour segments and pixel boundary;

creating a binary mask representative of said second region;

determining an area value of said closed loop comprising said two contour segments corresponding to an area value of the intersection of said first and second regions utilizing said area values and binary masks of said first and second regions; and

creating a binary mask representative of said closed loop comprising said two contour segments utilizing said binary masks of said first and second regions.

44. A method as claimed 38, wherein said step of combining incrementally said clockwise and counterclockwise closed loops comprises the sub-steps of:

determining an area value corresponding to an area value of a combination of a current said clockwise closed loop and previously combined clockwise closed loops

utilizing area values and binary masks of said current clockwise closed loop and previously combined clockwise closed loops;

creating a binary mask representative of said combination utilizing said binary masks of said current clockwise closed loop and previously combined clockwise closed

determining an area value corresponding to an area value of a combination of a current said counterclockwise closed loop and previously combined counterclockwise closed loops utilizing area values and binary masks of said current counterclockwise closed loop and previously combined counterclockwise closed loops; and

creating a binary mask representative of said combination utilizing said binary masks of said current counterclockwise closed loop and previously combined counterclockwise closed loops.

45. A method as claimed in claim 44, wherein said step of determining two winding count values representative of respective weighted averages of winding counts of said clockwise and counterclockwise closed loops comprises the sub-steps of:

summing said area values of each said clockwise closed loop to obtain said weighted average of said clockwise closed loops;

summing said area values of each said clockwise closed loop to obtain said weighted average of said clockwise closed loops.

46. A method as claimed in claim 38, wherein said step of determining the real opacity of the currently scanned pixel comprises the sub-step of:

determining a weighted average of winding counts for a region corresponding to an intersection of the combined clockwise and counterclockwise closed loops;

47. A method as claimed in claim 38, wherein said two weighted averages are determined incrementally and stored in respective two accumulators.

48. A method as claimed in claim 38, wherein the method comprises the step of:

approximating those line segments that lie within the currently scanned pixel with one or two line segments.

49. A method as claimed in claim 46, wherein the said real opacity of the currently scanned pixel is determined in accordance with the following formulae:

Opacity = $s_+f(|w_+|) + s_-f(|w_-|) + s_{-+}f(|w_+ - w_-|)$, where

$|w_+|$, $|w_-|$, $|w_+ - w_-|$ are the respective absolute values of the weighted averages of the winding counts of said combined clockwise closed loops, said combined counterclockwise closed loops, and said intersection of said combined clockwise and said combined counterclockwise closed loops, s_+, s_-, s_{-+}, are the respective area values of said combined clockwise closed loops, said combined counterclockwise closed loops, and said intersection of said combined clockwise and said combined counterclockwise closed loops, and said f() is a predetermined fill rule.

50. A method as claimed in claim 46, wherein the said real opacity of the currently scanned pixel is determined in accordance with the following formulae:

$f(s_+|w_+| + s_-|w_-| + s_{-+}|w_+ - w_-|)$, where

$|w_+|$, $|w_-|$, $|w_+ - w_-|$ are the respective absolute values of the weighted averages of the winding counts of said combined clockwise closed loops, said combined counterclockwise closed loops, and said intersection of said combined clockwise and said combined counterclockwise closed loops, s_+, s_-, s_{-+}, are the respective area values of said combined clockwise closed loops, said combined counterclockwise closed loops, and said intersection of said combined clockwise and said combined counterclockwise closed loops, and said f() is a predetermined fill rule.

51. A method as claimed in claim 46, wherein the said real opacity of the currently scanned pixel is determined in accordance with the following formulae:

$f((s_+|w_+|+ s_-|w_-| + s_{-+}|w_+ - w_-|) / (s_+ + s_- + s_{-+}))(s_+ + s_- + s_{-+})$, where $|w_+|$, $|w_-|$, $|w_+ - w_-|$ are the respective absolute values of the weighted averages of the winding counts of said combined clockwise closed loops, said combined counterclockwise closed loops, and said intersection of said combined clockwise and said combined counterclockwise closed loops, s_+, s_-, s_{-+}, are the respective area values of said combined clockwise closed loops, said combined counterclockwise closed loops, and said intersection of said combined clockwise and said combined counterclockwise closed loops, and said f() is a predctermined fill rule.

52. A method as claimed in claim 38, wherein the predetermined fill rule is an odd-even fill rule.

53. A method as claimed in claim 38, wherein the predetermined fill rule is a non-zero fill rule.

54. A method as claimed in claim 38, wherein the predetermined fill rule is a winding-counting fill rule.

55. A method as claimed in claim 49, 50, or 51, wherein the predetermined fill rule f(n) is determined in accordance with the following formulae:

$$f(n) = \begin{cases} n\alpha & n \le 1 \\ \alpha & n > 1 \end{cases} = \alpha \min(1, n)$$

where α is the intrinsic opacity and n is a number.

56. A method as claimed in claim 49, 50, or 51, wherein the predetermined fill rule f(n) is determined in accordance with the following formulae:

$$opacity = f(n) = m\alpha + (1 - (1 - \alpha)^{\lfloor n \rfloor})(1 - m\alpha)$$

where α is the intrinsic opacity and n is a number, $\lfloor n \rfloor$ is the largest integer less than or equal to n, and $m = n - \lfloor n \rfloor$.

57. A method as claimed in claim 49, 50, or 51, wherein the predetermined fill rule f(n) is determined in accordance with the following formulae:

$$f(n) = (n - \lfloor n \rfloor)\alpha \quad \lfloor n \rfloor \text{even}$$
$$= (1 - n + \lfloor n \rfloor)\alpha \quad \lfloor n \rfloor \text{odd}$$

where α is the intrinsic opacity and n is a number, $\lfloor n \rfloor$ is the largest integer less than or equal to n, and $m = n - \lfloor n \rfloor$.

58. A method of rendering a self-overlapping polygon in accordance with an odd-even fill rule, wherein the polygon is a set of one or more closed curves each comprising line segments, and the method performing, for a currently scanned pixel that overlaps both sides of a said line segment of the self-overlapping polygon within a currently scanned scanline, the steps of:

decomposing that portion of the polygon that lies within the currently scanned pixel into a number of closed loops comprising at least those portions of those line segments that lie within the currently scanned pixel, said closed loops are such that when they are combined the combination is substantially equivalent to that portion of the polygon that lies within the currently scanned pixel ;

combining incrementally said closed loops and determining a winding count value representative of a weighted average of winding counts of said closed loops, wherein said weighted average is effectively equivalent to the area of the combined loops;

determining a real opacity of the currently scanned pixel, where the real opacity of the currently scanned pixel is representative of the product of an intrinsic opacity of said polygon and said winding count value, and

rendering said currently scanned pixel with said determined real opacity.

59. Apparatus for rendering a self-overlapping polygon, wherein the polygon is a set of one or more closed curves each comprising line segments, and the apparatus

comprising means for processing a currently scanned pixel that overlaps both sides of a said line segment of the self-overlapping polygon within a currently scanned scanline, the processing means comprising:

means for decomposing that portion of the polygon that lies within the currently scanned pixel into a number of closed loops comprising at least those portions of those line segments that lie within the currently scanned pixel, said closed loops are such that when they are combined the combination is substantially equivalent to that portion of the polygon that lies within the currently scanned pixel ;

means for combining incrementally said closed loops and determining one or more winding count values representative of respective weighted averages of winding counts of said combined closed loops;

means for determining a real opacity of the currently scanned pixel according to a predetermined fill rule utilising an intrinsic opacity of said polygon and said one or more winding count values, and

means for rendering said currently scanned pixel with said determined real opacity.

60. Apparatus for rendering a self-overlapping polygon, wherein the polygon is a set of one or more closed curves each comprising line segments, and the apparatus comprising means for processing a currently scanned pixel that overlaps both sides of a said line segment of the self-overlapping polygon within a currently scanned scanline, the processing means comprising:

means for decomposing that portion of the polygon that lies within the currently scanned pixel into a number of clockwise or counterclockwise closed loops comprising at least those portions of those line segments that lie within the currently scanned pixel, said closed loops are such that when they are combined the combination is substantially equivalent to that portion of the polygon that lies within the currently scanned pixel;

means for combining incrementally said clockwise and counterclockwise closed loops respectively to produce two corresponding regions, and determining two winding count values representative of respective weighted averages of winding counts of said clockwise and counterclockwise closed loops;

means for determining a real opacity of the currently scanned pixel according to a predetermined fill rule utilising an intrinsic opacity of said polygon and said two winding count values, and

means for rendering said currently scanned pixel with said determined real opacity.

61. Apparatus for rendering a self-overlapping polygon in accordance with an odd-even fill rule, wherein the polygon is a set of one or more closed curves each comprising line segments, and the apparatus comprising means for processing a currently scanned pixel that overlaps both sides of a said line segment of the self-overlapping polygon within a currently scanned scanline, the processing means comprising:

means for decomposing that portion of the polygon that lies within the currently scanned pixel into a number of closed loops comprising at least those portions of those line segments that lie within the currently scanned pixel, said closed loops are such that when they are combined the combination is substantially equivalent to that portion of the polygon that lies within the currently scanned pixel ;

means for combining incrementally said closed loops and determining a winding count value representative of a weighted average of winding counts of said closed loops, wherein said weighted average is effectively equivalent to the area of the combined loops;

means for determining a real opacity of the currently scanned pixel, where the real opacity of the currently scanned pixel is representative of the product of an intrinsic opacity of said polygon and said winding count value, and

means for rendering said currently scanned pixel with said determined real opacity.

62. A computer program for rendering a self-overlapping polygon, wherein the polygon is a set of one or more closed curves each comprising line segments, and the computer program comprising code for processing a currently scanned pixel that overlaps both sides of a said line segment of the self-overlapping polygon within a currently scanned scanline, the processing code comprising:

code for decomposing that portion of the polygon that lies within the currently scanned pixel into a number of closed loops comprising at least those portions of those line segments that lie within the currently scanned pixel, said closed loops are such that when they are combined the combination is substantially equivalent to that portion of the polygon that lies within the currently scanned pixel ;

code for combining incrementally said closed loops and determining one or more winding count values representative of respective weighted averages of winding counts of said combined closed loops;

code for determining a real opacity of the currently scanned pixel according to a predetermined fill rule utilising an intrinsic opacity of said polygon and said one or more winding count values, and

code for rendering said currently scanned pixel with said determined real opacity.

63. A computer program for rendering a self-overlapping polygon, wherein the polygon is a set of one or more closed curves each comprising line segments, and the computer program comprising code for processing a currently scanned pixel that overlaps both sides of a said line segment of the self-overlapping polygon within a currently scanned scanline, the processing code comprising:

code for decomposing that portion of the polygon that lies within the currently scanned pixel into a number of clockwise or counterclockwise closed loops comprising at least those portions of those line segments that lie within the currently scanned pixel, said

closed loops are such that when they are combined the combination is substantially equivalent to that portion of the polygon that lies within the currently scanned pixel;

code for combining incrementally said clockwise and counterclockwise closed loops respectively to produce two corresponding regions, and determining two winding count values representative of respective weighted averages of winding counts of said clockwise and counterclockwise closed loops;

code for determining a real opacity of the currently scanned pixel according to a predetermined fill rule utilising an intrinsic opacity of said polygon and said two winding count values, and

code for rendering said currently scanned pixel with said determined real opacity.

64. A computer program for rendering a self-overlapping polygon in accordance with an odd-even fill rule, wherein the polygon is a set of one or more closed curves each comprising line segments, and the computer program comprising code for processing a currently scanned pixel that overlaps both sides of a said line segment of the self-overlapping polygon within a currently scanned scanline, the processing code comprising:

code for decomposing that portion of the polygon that lies within the currently scanned pixel into a number of closed loops comprising at least those portions of those line segments that lie within the currently scanned pixel, said closed loops are such that when they are combined the combination is substantially equivalent to that portion of the polygon that lies within the currently scanned pixel ;

code for combining incrementally said closed loops and determining a winding count value representative of a weighted average of winding counts of said closed loops, wherein said weighted average is effectively equivalent to the area of the combined loops;

code for determining a real opacity of the currently scanned pixel, where the real opacity of the currently scanned pixel is representative of the product of an intrinsic opacity of said polygon and said winding count value, and

code for rendering said currently scanned pixel with said determined real opacity.